# 百度语音识别 SDK iOS 版
# 开发手册

**V1.6.1**

**北京百度网讯科技有限公司**



北京百度网讯科技有限公司

# 目录

# 1 概念解释

对本文中将提到的概念约定如下：

- **语音识别**：被称为自动语音识别（Automatic Speech Recognition，ASR），其目标是将人类的语音中的词汇内容转换为计算机可读的输入，例如按键、二进制编码或者字符序列。
- **BDVRClient**：语音识别 SDK 的简称，详见下条。
- **语音识别 SDK**：即本开发包，文中简称为 BDVRClient。BDVRClient 是一个封装了语音采集、处理、网络收发等功能的语音识别解决方案。借助 BDVRClient 可以快速地在应用程序中集成语音识别功能。
- **应用程序**：在开发中使用了 BDVRClient，具有语音识别功能的产品线产品。
- **开发者**：在应用程序中使用 BDVRClient 的开发人员。

# 2 简介

百度语音识别 iOS 版本 SDK（以下简称 BDVRClient）是一种面向 iOS 移动设备的语音识别解决方案，以静态库方式提供。通过该方案，开发者可以轻松构建出功能丰富、交互性强的语音识别应用程序。

## 2.1 功能介绍

BDVRClient 支持下列功能：

基本功能：录音、语音数据处理、端点检测、网络通讯、状态通知、返回文字结果；

语音识别控件：集成提示音、音量反馈动效整套交互的对话框控件，方便开发者快速集成；

播放提示音：在录音前后播放提示音，优化用户体验；

监听语音音量：实时反馈用户当前说话声音能量强度；

语义理解：将语音识别成领域相关的语义结果。

本文档适用于对 i0S 应用开发有基本了解的开发人员。

## 2.2 兼容性

系统：支持 iOS 5.0 及以上系统。

架构：armv7、armv7s、arm64、i386、x86_64。

机型：iPhone 4+，iPad 2+和 iPod 5+。

硬件要求：需要有麦克风，用于支持语音录入。

网络：支持 NET、Wifi 网络环境。

## 2.3 开发包说明

表 1 开发包说明表

| 一级目录 | 二级目录 | 说明 |
|---|---|---|
| Headers | BDVoiceRecognitionClient.h | BDVRClient 无UI 头文件 |
| Headers | BDRecognizerViewController.h | BDVRClient UI 头文件 |
| Headers | BDRecognizerViewDelegate.h | BDVRClient UI 结果回调 |

| | | |
|---|---|---|
| | | 接口头文件 |
| Headers | BDRecognizerViewParamsObject.h | BDVRClient UI 中启动参数头文件 |
| Headers | BDTheme.h | BDVRClient UI 主题头文件 |
| Headers | BDVRRawDataRecognizer.h | BDVRClient 音频数据识别头文件 |
| Headers | BDVRFileRecognizer.h | BDVRClient 音频文件识别头文件 |
| BDVRClientSample | SDK Demo 源代码 | 开发示例 |
| Third-party | 各种第三方库 | 需要添加到项目中的第三方库 |
| libBDVoiceRecognitionClient<br>语音识别库静态库 | libBDVoiceRecognitionClient.a | 通用库，合并了真机 armv7、armv7s、arm64 和模拟器版的库。 |
| BDVoiceRecognitionClientResources<br>语音识别库资源文件 | Tone | 提示音资源文件 |
| | Theme | 识别控件主题 |
| Doc<br>使用文档 | 百度语音识别 iOS 版开发手册 | 开发者使用指南 |

## 2.4 总体框图

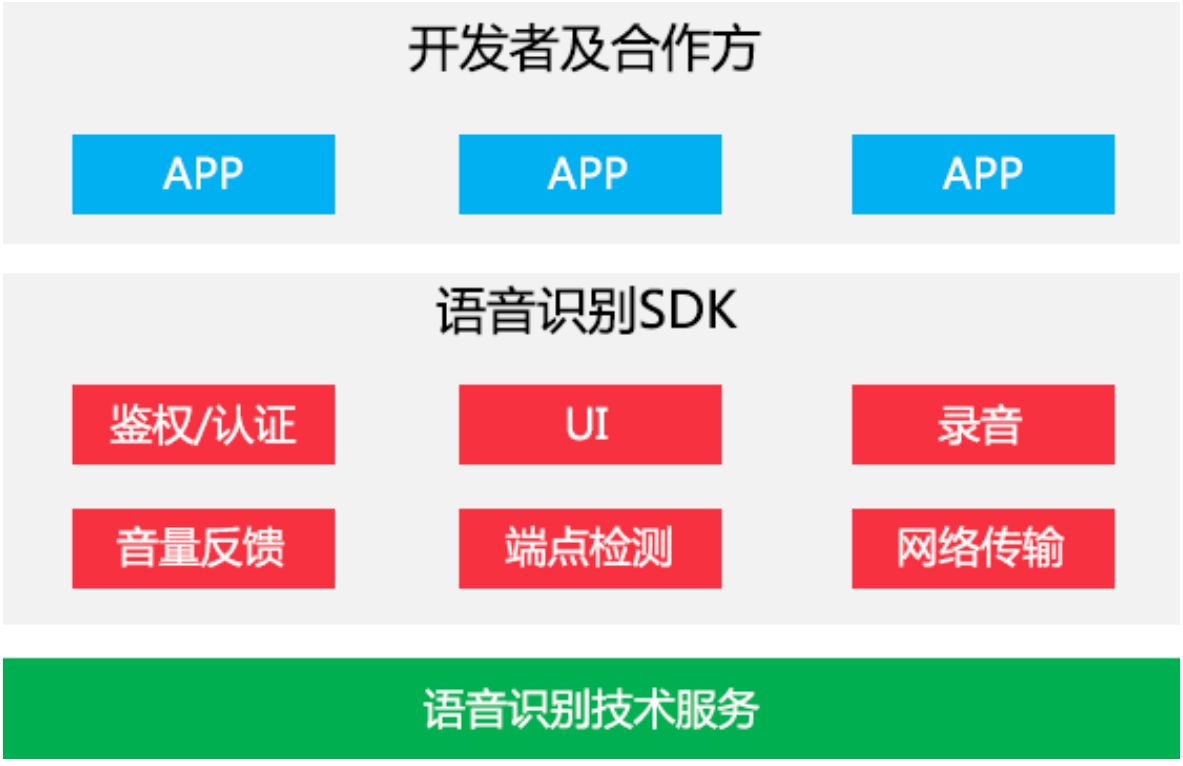


图 1 语音识别系统架构图

# 3 集成指南

从本章开始将进行 Step-By-Step 的讲解，介绍如何在应用工程中集成 BDVRClient。一个完整的 Demo 请参考开发包中的示例程序 BDVRClientSample。

## 3.1 注册开放云平台及创建应用

开发者需要在百度开放云平台 http://developer.baidu.com/ 注册帐号，并创建一个应用。

如图 2 红框所示，首次使用请先注册成为百度开放云平台的开发者。


图 2 注册开发者

注册成为开发者后，请选择首页右上角的“管理控制台”进入“开发者服务管理”。见图 3


图 3 进入管理控制台

创建应用：

（1） 点击“创建工程”。见图 4


图 4 创建应用

（2） 填写“应用名称”及其他信息，点击“创建”。见图 5


图 5 填写应用名称

（3） 应用创建成功。在应用基本信息页面，获取应用的 APPID、API Key、Secret Key 等信息。见图 6

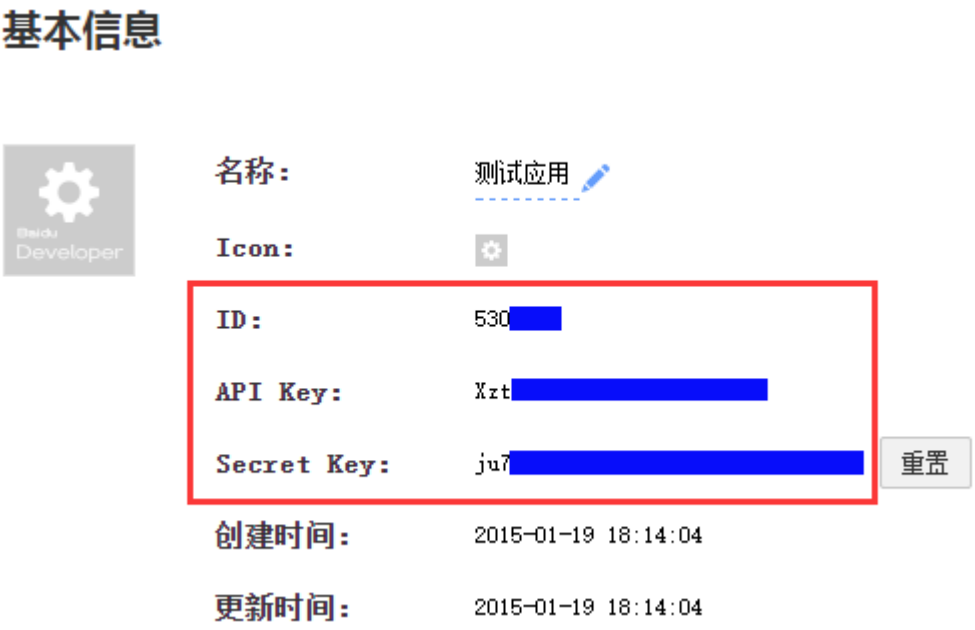


图 6 获取应用基本信息

## 3.2 申请开启语音识别服务

在使用语音识别服务之前，需要申请开启服务。您的申请提交后，后台工作人员会根据您的申请内容，进行批准或拒绝操作。审核需要 1-3 个工作日，审核的结果，会通过消息中心通知您。

进入应用，选择“媒体云”—“语音识别”，点击“申请开启服务”。根据提示填写使用场景、申请理由，提交申请。见图 7


图 7 申请开启语音识别服务

申请通过后，即拥有服务的使用权限。

## 3.3 引入编译需要的 Framework

BDVRClient 使用了录音和播放功能，因此需要在 Xcode 工程中引入 AudioToolbox.framework 和 AVFoundation.framework；BDVRClient 还使用到了网络状态检测功能，因此还需要引入 SystemConfiguration.framework；为了判断当前网络连接类型（2G/3G/4G），需要引入 CoreTelephony.framework；为了生成设备 UDID，需要引入 Security.framework；为了支持 gzip 压缩，需要引入 libz.1.dylib; 网络模块需要引入 CFNetwork.framework；某些场景需要获取设备地理位置以提高识别准确度，需引入 CoreLocation.framework。

为了支持识别控件，需要引入 OpenGLES.framework，QuartzCore.framework，GLKit.framework，CoreGraphics.framework 和 CoreText.framework。

添加方式：右键点击 Xcode 中的工程文件，在出现的界面中，选中 TARGETS 中应用，在出现的界面中选中 Build Phase->Link Binary With Libraries，点击界面中的“+”图标，在弹出的界面中选择此 7 个 Framework 即可，添加完成效果图如图 8 所示（libBDVoiceRecognitionClient.a 将在随后添加）。

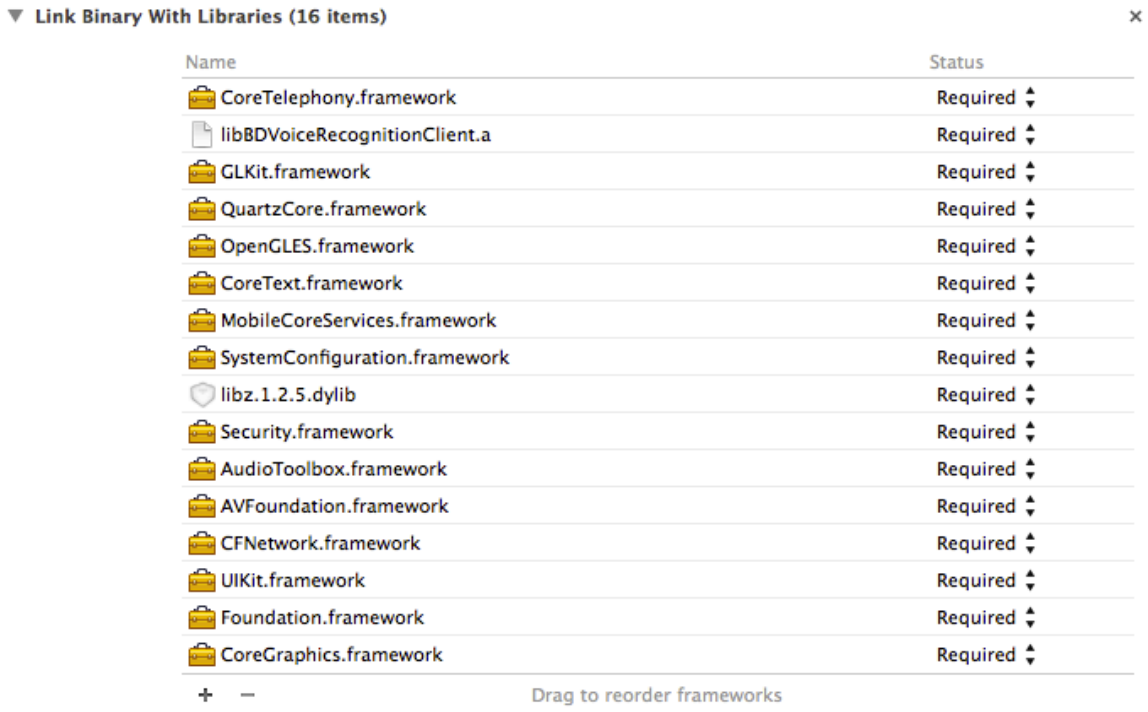


图 8 Framework 添加完成效果图

## 3.4 引入 BDVRClient 的头文件

首先将 BDVRClient 提供的头文件拷贝到工程目录下，在 XCode 中添加此文件，引入 BDVRClient 提供的头文件。

如果使用识别 UI，请添加如下头文件：

```
#import "BDRecognizerViewController.h"

#import "BDRecognizerViewDelegate.h"
```

如果只使用识别接口，添加如下头文件：

```
#import "BDVoiceRecognitionClient.h"
```

如果要对音频数据或音频文件直接进行识别，请分别添加如下头文件：

```
#import "BDVRRawDataRecognizer.h"

#import "BDVRFileRecognizer.h"
```

## 3.5 引入静态库文件

BDVRClient 提供了模拟器 5.0 及更新版本，真机 armv7、armv7s 和 arm64 四种环境所使用的静态库文件，分别存放在开发包的 libBDVoiceRecognitionClient 文件夹下，详细见“开发包说明”部分。

引入前准备：静态库中采用 Objective C++实现，因此需要保证工程中引用静态库头文件的实现文件的扩展名必须为.mm。

引入静态库文件的具体方式是：将 libBDVoiceRecognitionClient 文件夹下的 libBDVoiceRecognitionClient.a 采用添加文件方式添加到工程的 Framework 目录下，添加完成效果如图 9 所示。

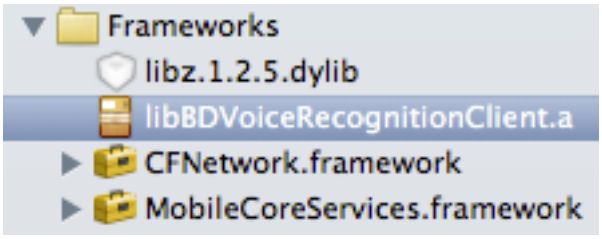


图 9 静态库添加完成效果图

说明：libBDVoiceRecognitionClient.a 是采用 lipo 命令将 armv7，armv7s，arm64 和模拟器 Debug 版的.a 合并成的一个通用的.a 文件，避免开发者在 build 不同 target 时频繁替换.a 文件的问题。

## 3.6 添加第三方开源库

BDVRClient 中使用了第三方开源库，包括 JSONKit、TTTAttributedLabel 和苹果非官方的录音 API，如果产品项目中已经包含其中的部分库，请勿重复添加，否则，请添加这三种第三方开源库到项目中，第三方库文件可以在 SDK 开发包下的 Third-party 目录下找到。

注意：其中第三方库 TTTAttributedLabel 需要设置为 ARC 方式编译。

## 3.7 引入库所需的资源文件

将开发包中的 BDVoiceRecognitionClientResources 文件夹加入到工程中，具体的添加办法，将文件夹拷入工程文件中，以“create folder references for any adds”方式添加到工程的资源 Group 中，

添加方式如下图 10 所示。

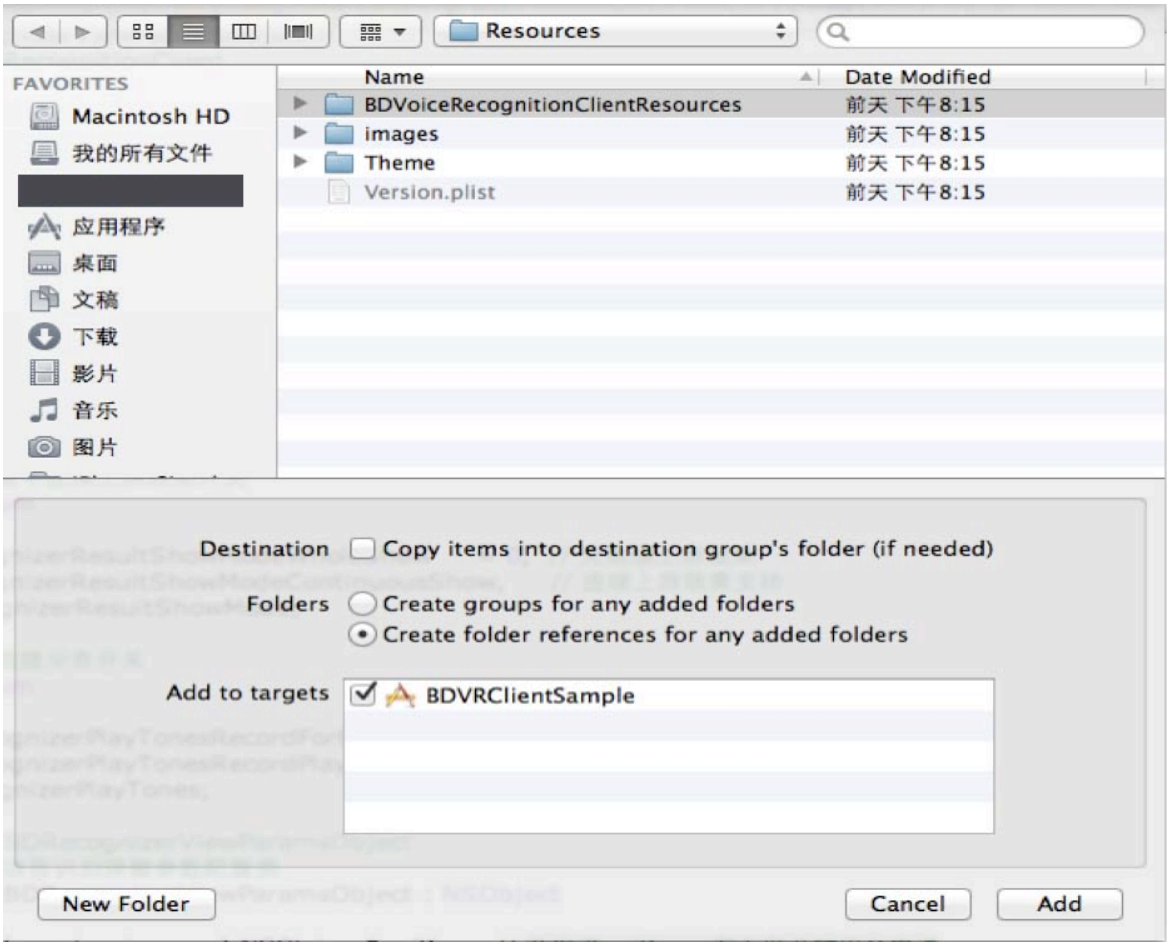


图 10 BDVoiceRecognitionClientResources 添加方式图

# 4 语音识别

百度语音 SDK 提供英语、粤语、普通话识别，支持输入、热词、地图、音乐、应用、web、购物、健康、打电话和视频等 10 种不同的应用场景。其中搜索、地图、音乐等模式适合短 Query 的输入场景，在识别过程中会将语气词、标点等过滤，尾点检测会更灵敏。输入模式适合短信输入等长句场景，尾点检测会迟钝一些，可以进行连续多句识别。开发者可以通过调用语音识别控件快速集成，也可以使用底层 API 接口自行设计语音交互。

SDK 还支持语义理解能力，可以将用户的语音直接转换成需求意图。语义具有领域性特征，不属于任何领域的语义是不存在的。同样的语言，在不同的领域中所代表的含义可能截然不同。语义理解就是把语言在特定领域所代表语义通过计算机可处理的表示方式理解出来。具体支持的领域及数据格式请参考《百度语义理解协议》。

如果开启了语义解析能力，结果将做为 JSON 字符串放至在候选数组的首个元素。JSON 对象的结构如下：

| 字段名 | 数据类型 | 描述 |
| --- | --- | --- |
| **item** | JSONArray<String> | 为 String 类型的 JSONArray，存储语音识别结果 |

| | | |
|---|---|---|
| **json_res** | String | 语义解析结果，为 Json 格式的 String。 |

json_res 转换成 Json 对象结构如下，详细参考《百度语义理解协议》

| 字段名 | 数据类型 | 描述 |
|---|---|---|
| **raw_text** | string | 原始文本 |
| **parsed_text** | string | 分词结果 |
| **results** | JSONArray | 语义理解意图数组，可能为空 |

## 4.1 语音识别控件

BDVRClient 提供了语音识别控件 BDRecognizerDialogController，并提供了更换主题功能，方便开发者快速引入语音识别功能。

### 4.1.1 创建识别控件对象

创建方式如下：

```
BDRecognizerViewController  *recognizerViewController = [[BDRecognizerViewControlle
r alloc] initWithOrigin:CGPointMake(9, 128) withTheme: [BDTheme defaultTheme]];

recognizerViewController.enableFullScreenMode = NO; //设置是否全屏模式
```

#### 4.1.1.1 识别控件起始位置

上述接口中，第一个参数为识别控件的在屏幕中的起始位置，识别控件会添加到当前的 keyWindow 上。

#### 4.1.1.2 识别控件主题

上述接口中，第二个参数为识别控件的主题，如果参数为空，则为默认主题，主题可以从下面接口中任选其一，当选择为某一个主题，则将开发包中 BDVoiceRecognitionClientResources/Theme 中对应的主题 Bundle 添加到项目资源目录中。

生成一个主题的方式如下：

```
BDTheme *theme=[BDTheme defaultTheme]; // 获取默认皮肤
```

目前 BDVRClient 的主题分为亮和暗两种系列，各分为红，橙，蓝，绿四种，开发者可以自行选用，具体选择的接口如下：

```
+ (instancetype)defaultTheme;
```

```
+ (instancetype)lightBlueTheme;
+ (instancetype) darkBlueTheme
+ (instancetype) lightGreenTheme
+ (instancetype) darkGreenTheme
+ (instancetype) lightOrangeTheme
+ (instancetype) darkOrangeTheme
+ (instancetype) lightRedTheme
+ (instancetype) darkRedTheme
```

说明：因为程序一般只需要一套皮肤，因此开发者在使用不用将 Theme 文件中的所有 Bundle 都拷入工程。

### 4.1.2 启动识别

创建识别控件后，启动方法如下：

```
[recognizerViewController startWithParams:paramsObject];
```

其中启动时需要设置识别参数信息，即类 BDRecognizerViewParamsObject，此参数不能为空，否则将返回 NO，即启动识别失败，可设置的参数参考表 2。

表 2：启动参数配置表

| 参数名称 | 参数类型 | 默认值 | 描述 |
|---|---|---|---|
| **apiKey** | NSString | | 开放云平台认证 API_key |
| **secretKey** | NSString | | 开放云平台认证 Secret_key |
| **language** | TBVoiceRecognitionLanguage | LANGUAGE_CHINESE | 语种，见 BDRecognizerViewParamsObject.h 中定义的枚举 |
| **isNeedNLU** | BOOL | false | 是否回传语义理解结果，仅在搜索模式下有效 |
| **recogPropList** | NSArray | | 领域参数，取值参见 BDVoiceRecognitionClient.h 中对应枚举类型 TBVoiceRecognitionResourceType |

| | | | |
|---|---|---|---|
| **cityID** | NSInteger | 1(标示为全国) | 城市标识，仅当领域参数包含EVoiceRecognitionPropertyMap时有效 |
| **disablePuncs** | BOOL | false | 是否禁用标点 |
| **resultShowMode** | TBDRecognizerResultShowMode | BDRecognizerResultShowModeContinuousShow | 设置识别结果显示方式，分为连续上屏和无连续上屏效果，取值参见BDRecognizerViewParamsObject.h 中定义的枚举 |
| **recordPlayTones** | TBDRecognizerPlayTones | EBDRecognizerPlayTonesRecordPlay | 提示音开关，取值参见BDRecognizerViewParamsObject.h 中定义的枚举 |
| **tipsTitle** | NSString | | 提示语列表界面标题 |
| **tipsList** | NSArray | | 提示语列表，用于提示用户说出的话 |
| **isShowTipsOnStart** | BOOL | YES | 是否在第一次启动时打开提示语列表 |
| **isShowTipAfter3sSilence** | BOOL | YES | 是否在用户 3 秒未说话时浮出随机提示语 |
| **isShowHelpButtonWhenSilence** | BOOL | YES | 静音超时后将“取消”按钮替换为“帮助” |
| **isNeedVad** | BOOL | YES | 是否需要进行语音端点检测 |
| **isNeedCompress** | BOOL | YES | 是否对上传音频进行压缩 |

说明：apiKey 和 secretKey 不能为非法值，否则识别不能正常进行。

### 4.1.3 识别结果接受

使用识别控件进行识别时，为了能正确的接受识别结果，必须实现BDRecognizerViewDelegate头文件中的BDRecognizerViewDelegate协议，协议实现示例如下：

```
-(void)onEndWithViews:(BDRecognizerViewController *)aBDRecognizerViewController wi
thResults:(NSArray *)aResults
{
```

```
    if ([[BDVoiceRecognitionClient sharedInstance] getRecognitionProperty] != EVoice
RecognitionPropertyInput)
    {
        // 在非输入模式下，当 isNeedNLU = NO 时，结果是一个候选句列表，为 NSArray，
        // 元素是 NSString，例如["公园", "公元"]
        // 当 isNeedNLU = YES 时，结果类型仍为 NSArray，只有一个 NSString 元素，
        // 内容是 JSON 串，开发者需要自行解析，解析方法请参考语义解析文档
        NSMutableString *tmpString = [aResults objectAtIndex:0];
    }
    else
    {
        //按照词粒度改错，并且返回可信度。结果是一个 NSArray，但元素仍然是 NSArray，
        // 此元素就是词语的解码和改错结果，其元素为 NSDictionary（key 为候选词，value 为
        // 可信度）
        // 输入模式下的结果为带置信度的结果，示例如下：
        // [
        //     [
        //         {
        //             "百度" = "0.6055192947387695";
        //         },
        //         {
        //             "摆渡" = "0.3625582158565521";
        //         },
        //     ]
        //     [
        //         {
        //             "一下" = "0.7665404081344604";
        //         }
        //     ],
        // ]
    }
}
```

### 4.1.4 获取录音数据

如果需要记录录音数据，实现 BDRecognizerViewDelegate 的 onRecordDataArrived:方法即可，当该方法被调用时，将传回包含录音数据的 NSData 对象，通过对该数据的处理可以得到录音数据。

注：传回的录音数据是未经压缩的 PCM 数据，采样位数为 16bit，采样率根据网络情况的不同而不同，当前网络情况为 WiFi 时采样率为 16k，否则为 8k。将返回的数据顺序拼接起来可以得到完整的音频文件。

## 4.2 API 方式

BDVRClient 在提供识别控件的同时，提供识别的无 UI 接口，更加方便开发者的自定义开发，

使用接口进行语音识别方法如下。

### 4.2.1 获取语音识别单例

BDVRClient 的设计思路是采用单例模式，获得单例代码如下：

```
BDVoiceRecognitionClient*sharedInstance=[BDVoiceRecognitionClient sharedInstance];
```

### 4.2.2 设置语音识别模式

当前支持 18 种不同的识别垂类：包括地图、音乐、视频、应用、web、热词、购物、健康、打电话、医疗、汽车、娱乐餐饮、财经、游戏、菜谱、助手、手机充值和输入法，默认为输入 EVoiceRecognitionPropertyInput，取值参见 BDVoiceRecognitionClient.h 中的枚举类型 TBDVoiceRecognitionProperty。除 EVoiceRecognitionPropertyInput 和 EVoiceRecognitionPropertySong 只能单独使用外，其他识别领域可以一次设置多个，同时放入数组中，cityID 参数仅当识别垂类中包含 EVoiceRecognitionPropertyMap 时有效，cityID 为 1 表示地图全搜，[2~10000]表示地图分区搜索。

请根据应用的语音识别需求选择对应的信息设置，如果选择垂类信息不恰当会导致语音识别准确率的异常。同时请谨慎选择垂类的范围，垂类选择过多也会导致语音识别准确率有一定幅度的降低。

```
BOOL res = [[BDVoiceRecognitionClient sharedInstance] setPropertyList: @[[NSNumber
numberWithInt: EVoiceRecognitionPropertyMap]]];

res = [[BDVoiceRecognitionClient sharedInstance] setCityID: 1];
```

### 4.2.3 设置识别语言

当前支持 3 种不同的识别语言：普通话、粤语和美式英文，默认为普通话，取值参见 BDVoiceRecognitionClient.h 中的枚举类型。

```
[[BDVoiceRecognitionClient sharedInstance] setLanguage: EVoiceRecognitionLanguageEn
glish];
```

### 4.2.4 其他设置

可通过如下方法禁用标点符号，默认为不禁用。

```
 [[BDVoiceRecognitionClient sharedInstance] disablePuncs:YES];
```

设置是否对语音进行端点检测，默认开启。如进行端点检测，SDK 会自动判断说话是否结束，从而停止录音；否则，需要调用 speakFinish 显式结束识别录音。

```
[[BDVoiceRecognitionClient sharedInstance] setNeedVadFlag:YES];
```

设置是否对上传的语音进行压缩，默认开启。压缩语音会节约手机流量，但是会消耗 CPU 资源。

```
[[BDVoiceRecognitionClient sharedInstance] setNeedCompressFlag:YES];
```

### 4.2.5 开始语音识别

1. 开始语音识别需要配置参数，具体配置可参考开发包头文件中的说明。开发者信息 ApiKey 和 SecretKey 为必须配置项，其余可选。设置方式如下：

```
 [[BDVoiceRecognitionClient sharedInstance] setApiKey:@"apiKey " withSecretKey:@”secretKey”];
```

2. 启动语音识别：此操作将会启动录音，预处理和网络通讯三个模块的工作。

```
int startStatus = [[BDVoiceRecognitionClient sharedInstance] startVoiceRecognition:self];

    switch(startStatus)
    {
        case EVoiceRecognitionStartWorking :..... // 启动成功
    .....
    }
```

注意：

startVoiceRecognition 有可能启动失败，原因有无网络，无麦克风，或者没有实现 delegate 等等，识别的原因参考表 3。

表 3：启动识别时返回的错误说明

| 错误名 | 描述 |
| --- | --- |
| EVoiceRecognitionStartWorkNOMicrophonePermission | 没有系统麦克风使用权限，iOS7 系统 |
| EVoiceRecognitionStartWorkNoAPIKEY | 没有 ApiKey，需要验证开放云平台的注册信息 |
| EVoiceRecognitionStartWorkGetAccessTokenFailed | 获取有效的 AccessToken 错误，需要验证开放云平台的注册信息。 |
| EVoiceRecognitionStartWorkNetUnusable | 没有网络 |

| | |
|---|---|
| EVoiceRecognitionStartWorkDelegateInvaild | 没有实现语音回调接口 |
| EVoiceRecognitionStartWorkRecorderUnusable | 录音设备不可用 |
| EVoiceRecognitionStartWorkPreModelError | 启动预处理模块出错 |
| EVoiceRecognitionStartWorkPropertyInvalid | 识别属性或城市 ID 设置不合法 |

### 4.2.6 接收语音识别结果

接收结果需要实现 MVoiceRecognitionClientDelegate 协议。此协议主要是通知识别过程中的各种状态和结果，包括三个方法，详细请参见开发包中的 BDVoiceRecognitionClient 头文件。

语音输入和语音搜索相比，识别从单句变为多句，因此，其结果返回不止一次。BDVRClient 中在 VoiceRecognitionClientWorkStatus:(int)aStatus obj:(id)aObj 方法中添加了一个新的状态 EVoiceRecognitionClientWorkStatusReceiveData，用来接口输入模式下返回的子句。

注意：识别结果的接口中在输入模式下返回的识别结果数据结构与搜索不同。搜索返回结果是按照句子粒度改错，是一个候选句列表，为 NSArray，元素是 NSString；输入返回的结果是按照词粒度改错，并且返回可信度。结果是一个 NSArray，但元素仍然是 NSArray，此元素就是词语的解码和改错结果，其元素为 NSDictionary（key 为候选词，value 为可信度）。

1. 应用程序如果只是实现简单语音识别功能，只需要实现如下方法即可，具体实现示例如下：

```
- (void)VoiceRecognitionClientWorkStatus:(int)aStatus obj:(id)aObj
{
    switch (aStatus)
    {
        case EVoiceRecognitionClientWorkStatusFlushData:
        {
            // 该状态值表示服务器返回了中间结果，如果想要将中间结果展示给用户（形成连续上屏的效果），
            // 可以利用与该状态同时返回的数据，每当接到新的该类消息应当清空显示区域的文字以免重复
            NSMutableString *tmpString = [[NSMutableString alloc] initWithString:@""];
            [tmpString appendFormat:@"%@",[aObj objectAtIndex:0]];
            NSLog(@"result: %@", tmpString);
            break;
        }
        case EVoiceRecognitionClientWorkStatusFinish:
        {
           // 该状态值表示语音识别服务器返回了最终结果，结果以数组的形式保存在 aObj 对象中
           // 接受到该消息时应当清空显示区域的文字以免重复
            if ([[BDVoiceRecognitionClient sharedInstance] getRecognitionProperty] != EVoiceReco
gnitionPropertyInput)
            {
               NSMutableArray *resultData = (NSMutableArray *)aObj;
               NSMutableString *tmpString = [[NSMutableString alloc] initWithString:@""];

                // 获取识别候选词列表
               for (int i=0; i<[resultData count]; i++)
               {
                  [tmpString appendFormat:@"%@\r\n",[resultData objectAtIndex:i]];
               }

                NSLog(@"result: %@", tmpString);
                [tmpString release];
```

```
        }
        else
        {
          NSMutableString *sentenceString = [[NSMutableStringalloc] initWithString:@""];
          for (NSArray *result inaObj)// 此时 aObj 是 array，result 也是 array
          {
              // 取每条候选结果的第一条，进行组合
              // result 的元素是 dictionary，对应一个候选词和对应的可信度
            NSDictionary *dic = [result objectAtIndex:0];
            NSString *candidateWord = [[dic allKeys] objectAtIndex:0];
              [sentenceString appendString:candidateWord]
          }
          NSLog(@"result: %@", sentenceString);
        }

        break;
    }
    case EVoiceRecognitionClientWorkStatusReceiveData:
    {
         // 此状态只在输入模式下发生，表示语音识别正确返回结果，每个子句会通知一次（全量，
         // 即第二次收到该消息时所携带的结果包含第一句的识别结果），应用程序可以
         // 逐句显示。配合连续上屏的中间结果，可以进一步提升语音输入的体验
         NSMutableString *sentenceString = [[NSMutableStringalloc] initWithString:@""];
         for (NSArray *result inaObj)// 此时 aObj 是 array，result 也是 array
         {
             // 取每条候选结果的第一条，进行组合
             // result 的元素是 dictionary，对应一个候选词和对应的可信度
           NSDictionary *dic = [result objectAtIndex:0];
           NSString *candidateWord = [[dic allKeys] objectAtIndex:0];
             [sentenceString appendString:candidateWord]
         }
         NSLog(@"result: %@", sentenceString);
        break;
    }
    case EVoiceRecognitionClientWorkStatusNewRecordData:
    {
         // 有音频数据输出，音频数据格式为 PCM，在有 WiFi 连接的条件下为 16k16bit，非 WiFi
         // 为 8k16bit
        break;
    }
    case EVoiceRecognitionClientWorkStatusEnd:
    {
        // 用户说话完成，但服务器尚未返回结果
        break;
    }
    case EVoiceRecognitionClientWorkStatusCancel:
    {
        // 用户主动取消
        break;
    }
    case EVoiceRecognitionClientWorkStatusError:
    {
        // 错误状态
        break;
    }
    case EVoiceRecognitionClientWorkPlayStartTone:
    case EVoiceRecognitionClientWorkPlayStartToneFinish:
```

```
        case EVoiceRecognitionClientWorkStatusStartWorkIng:
        case EVoiceRecognitionClientWorkStatusStart:
        case EVoiceRecognitionClientWorkPlayEndToneFinish:
        case EVoiceRecognitionClientWorkPlayEndTone:
        case EVoiceRecognitionClientWorkStatusEndAndToTackNetWork:
        {
            // 其他中间状态
             break;
        }
    }
}
```

2. 如果应用程序需要监听 BDVRClient 识别过程中的网络状态，需要实现如下方法，代码如下：

```
- (void)VoiceRecognitionClientNetWorkStatus:(int) aStatus
{
    switch (aStatus)
    {
        case EVoiceRecognitionClientNetWorkStatusStart:
        {
            // 底层网络通讯模块通讯开始，用户可以在状态条中显示菊花转等等，提示用户
            break;
        }
        case EVoiceRecognitionClientNetWorkStatusEnd:
        {
            // 底层网络通讯模块通讯完成，同样此状态可以在 UI 上给用户提示
            break;
        }
    }
}
```

3. BDVRClient 识别中发生错误后，如果应用程序需要得到更加详细的错误信息，需要实现如下方法，具体的错误类型说明参考表 4。

表 4：语音识别错误说明

| ErrorType | 描述 | ErrorCode(值) | 描述 |
|---|---|---|---|
| EVoiceRecognitionClientErrorStatusClassVDP | 客户端异常 | EVoiceRecognitionClientErrorStatusUnKnow (1101) | 未知 |
| | | EVoiceRecognitionClientErrorStatusNoSpeech (1102) | 没有说话 |
| | | EVoiceRecognitionClientErrorStatusShort (1103) | 语音太短 |
| | | EVoiceRecognitionClientErrorStatusException (1104) | SO 异常 |
| EVoiceRecognitionClientErrorStatusClassRecord | 录音设备异常 | EVoiceRecognitionClientErrorStatusChangeNotAvailable(1201) | 设备不可用 |
| | | EVoiceRecognitionClientErrorStatusIntrerrupti | 录音中 |

| | | | |
|---|---|---|---|
| | | on(1202) | 断 |
| EVoiceRecognitionClientErrorStatusClassLocalNet | 网络异常 | EVoiceRecognitionClientErrorNetWorkStatusUnusable(1301) | 网络不可用 |
| | | EVoiceRecognitionClientErrorNetWorkStatusError(1302) | 连接失败 |
| | | EVoiceRecognitionClientErrorNetWorkStatusTimeOut(1303) | 网络超时 |
| | | EVoiceRecognitionClientErrorNetWorkStatusParseError(1304) | 数据解析失败 |
| EVoiceRecognitionClientErrorStatusClassServerNet | 服务器错误 | EVoiceRecognitionClientErrorNetWorkStatusServerParamError(3001) | 参数错误 |
| | | EVoiceRecognitionClientErrorNetWorkStatusServerRecognError(3002) | 识别错误 |
| | | EVoiceRecognitionClientErrorNetWorkStatusServerNoFindResult(3003) | 没有匹配结果 |
| | | EvoiceRecognitionClientErrorNetWorkStatusServerAppNameUnknownError(3004) | 认证失败 |
| | | EEVoiceRecognitionClientErrorNetWorkStatusServerSpeechQualityProblem(3005) | 声音不符合标准 |
| | | EVoiceRecognitionClientErrorNetWorkStatusServerUnknownError(3006) | 未知错误 |

示例代码如下：

```
- (void)VoiceRecognitionClientErrorStatus:(int) aStatus subStatus:(int)aSubStatus
{
    switch (aStatus)
    {
        case EVoiceRecognitionClientErrorStatusClassVDP:
        {
            //语音数据处理出错
            switch (aSubStatus)
            {
                case EVoiceRecognitionClientErrorStatusNoSpeech:
                {
                    //用户未说话
                    break;
                }
                //其他 case
            }
            break;
        }
        case EVoiceRecognitionClientErrorStatusClassRecord:
        {
            // 录音出错
```

```
            break;
        }
        case EVoiceRecognitionClientErrorStatusClassLocalNet:
        {
            // 网络出错
            break;
        }
        case EVoiceRecognitionClientErrorStatusClassServerNet:
        {
            // 服务器返回错误
            break;
        }
}
```

### 4.2.7 用户取消操作

BDVRClient 提供用户取消接口，响应用户取消操作，安全释放相关资源。实现代码说明如下：

```
-   (void)cancel:(id)sender
{
    //停止 BDVRClient 的识别过程，该方法会释放相应的资源，并向接受识别结果接口中发送用户取消通知。
    [[BDVoiceRecognitionClientsharedInstance] stopVoiceRecognition];
}
```

注意：

调用完此接口后，应该在接受识别结果接口中实现对应取消状态的处理，比如取消 UI 等等。

### 4.2.8 用户主动结束识别

BDVRClient 支持智能检测用户是否说话完毕，但在噪声干扰比较严重的环境下，智能检测的效果可能会非常不好，因此，BDVRClient 支持通过外部调用来主动结束识别。如果想让程序有更好的用户体验，应当为用户提供主动结束识别的功能，例如：在程序的界面中提供一个“说完了”的按钮，让用户主动结束识别。主动结束的实现代码如下：

```
-   (void)finish:(id)sender
{
    // 结束语音识别，录音完成，此后可以放心地等待结果返回和状态通知，不需要添加额外代码。
    [[BDVoiceRecognitionClientsharedInstance] speakFinish];
}
```

### 4.2.9 请求自然语言理解结果

#### 4.2.9.1 开启自然语言理解功能

```
    [[BDVoiceRecognitionClient sharedInstance] setConfig:@”nlu” withFlag:YES];
```

#### 4.2.9.2 解析自然语言理解结果

识别结果类型仍为 NSArray，但只有一个 NSString 元素，内容是 JSON 串，开发者需要自行解析，解析方法请参考语义解析文档。

## 4.2.10 播放提示音

语音识别过程中，可以提醒用户开始说话和说话结束，以便能使识别的结果更加符合用户的预期。BDVRClient 支持的程序中播放提示音，来提示用户开始说话和结束说话。提示音分为开始说话提示音和结束说话提示音。BDVRClient 默认不播放提示音，需要支持此功能步骤如下.

### 4.2.10.1 开启播放开始说话提示音

1. 引入提示音文件

如果使用 BDVRClient 提供的默认的文件，则可以忽略本步骤。如果需要添加自己的文件，则需要覆盖 BDVoiceRecognitionClientResources 目录下的 Tone 文件夹中的 record_start.caf 文件。

2. 打开播放提示音的开关

在开始语音识别前，配置提示音的开关：

```
// EVoiceRecognitionPlayTonesRecStart 为开关类型
BOOL res = [[BDVoiceRecognitionClient sharedInstance] setPlayTone:EVoiceRecognition
PlayTonesRecStartisPlay:YES ];

if (res)
{
    // 设置成功
}
```

### 4.2.10.2 开启播放完成说话提示音

1. 引入提示音文件

与播放开始说话提示音相似，只是语音文件为 record_end.caf。

2. 打开播放提示音的开关

在开始语音识别前，进行配置开关，

```
// EVoiceRecognitionPlayTonesRecEnd 为开关类型
[[BDVoiceRecognitionClient sharedInstance] setPlayTone:EVoiceRecognitionPlayTonesR
ecEndisPlay:YES ];
```

注意：

使用自己的语音文件格式必须是 caf，并且文件名和路径不要做修改，否则 BDVRClient 无法进行播放。

建议：

提示音的时长不要超过 0.3 秒。

## 4.2.11 监听语音音量

BDVRClient 提供监听语音音量功能，通过此功能，可以获取到用户当前说话的音量级别（0-100），由此来判断是否产生截副通知用户，或者也可以把获取的音量级别以某种 UI 方式（比

如柱状图）呈现来提醒用户说话声音不要太大或太小。BDVRClient 默认不开启此功能，具体的详细实现流程如下。

#### 4.2.11.1 打开监听语音音量功能

如果要使用此功能，需要在一次语音识别开始前设置。

```
[[BDVoiceRecognitionClient sharedInstance] listenCurrentDBLevelMeter];
```

#### 4.2.11.2 获取语音音量级别

当开启监听语音音量级别后，需要开发者在程序中主动获取音量级别，代码如下：

```
[[BDVoiceRecognitionClient sharedInstance] getCurrentDBLevelMeter];
```

注意：

BDVRClient 语音音量级别更新的频率是 10 次/s。

建议：

在开始录音后进行获取，也就是收到 EVoiceRecognitionClientWorkStatusStartWorkIng 通知后开发获取。

#### 4.2.11.3 取消监听语音音量功能

BDVRClient 采用单例模式，如果一旦不使用时要取消此功能，这样可以减少识别程序不必要的开销，取消时机在一次语音识别开始之前。代码如下：

```
[[BDVoiceRecognitionClient sharedInstance] cancelListenCurrentDBLevelMeter];
```

### 4.2.12 获取录音数据

如果需要记录录音数据，可以在 VoiceRecognitionClientWorkStatus 中监听 EVoiceRecognitionClientWorkStatusNewRecordData 消息，当该消息到达时，aObj 对象将是 NSData 类，通过对该数据的处理可以得到录音数据。

注：传回的录音数据是未经压缩的 PCM 数据，采样位数为 16bit，采样率根据网络情况的不同而不同，当前网络情况为 WiFi 时采样率为 16k，否则为 8k。将返回的数据顺序拼接起来可以得到完整的音频文件。

### 4.2.13 音频数据识别

BDVRClient 还支持直接对音频数据或者音频文件进行识别，目前支持 16k-16bit 和 8k-16bit 的 pcm 格式语音数据识别。

发起音频文件识别的示例代码如下：

```
    NSBundle *bundle = [NSBundle mainBundle];
    NSString* recordFile = [bundle pathForResource:@"example_localRecord"
                                            ofType:@"pcm" inDirectory:nil];
```

```
    self.fileRecognizer = [[BDVRFileRecognizer alloc]
           initFileRecognizerWithFilePath:recordFile sampleRate:16000
           mode:EVoiceRecognitionPropertyInput delegate:self];

    int status = [self.fileRecognizer startFileRecognition];
    if (status != EVoiceRecognitionStartWorking) {
        NSLog([NSString stringWithFormat:@"错误码：%d\r\n", status]);
        return;
    }
```

发起音频数据识别的示例代码如下（仍然以读文件为例）：

```
    self.rawDataRecognizer = [[BDVRRawDataRecognizer alloc]
            initRecognizerWithSampleRate:16000
                                    mode:EVoiceRecognitionPropertyInput
                                delegate:self];
    int status = [self.rawDataRecognizer startDataRecognition];

    if (status != EVoiceRecognitionStartWorking) {
        NSLog([NSString stringWithFormat:@"错误码：%d\r\n", status]);
        return;
    }
    NSThread* fileReadThread = [[NSThread alloc] initWithTarget:self
                                          selector:@selector(fileReadThreadFunc)
                                            object:nil];
    [fileReadThread start];
```

其中的读文件线程方法如下：

```
- (void)fileReadThreadFunc
{
    NSBundle *bundle = [NSBundle mainBundle];
    NSString* recordFile = [bundle pathForResource:@"example_localRecord"
                                            ofType:@"pcm" inDirectory:nil];

    int hasReadFileSize = 0;

    // 每次向识别器发送的数据大小，建议不要超过 4k，
    // 这里通过计算获得：采样率 * 时长 * 采样大小 / 压缩比
    // 其中采样率支持 16000 和 8000，采样大小为 16bit，压缩比为 8，时长建议不要超过 1s
    int sizeToRead = 16000 * 0.080 * 16 / 8;
    while (YES) {
        NSFileHandle *fileHandle = [NSFileHandle
                                    fileHandleForReadingAtPath:recordFile];
        [fileHandle seekToFileOffset:hasReadFileSize];
        NSData* data = [fileHandle readDataOfLength:sizeToRead];
        [fileHandle closeFile];
        hasReadFileSize += [data length];
        if ([data length]>0)
        {
            [self.rawDataRecognizer sendDataToRecognizer:data];
        }
        else
        {
            [self.rawDataRecognizer allDataHasSent];
```

```
            break;
        }
    }
}
```

完整代码请参见开发包所附示例 BDVRClientSample。

## 4.3 自定义识别结果

开发者可以根据自己的需求上传相关数据，使得语音识别结果更加精确，当前支持联系人上传。

### 4.3.1 联系人

**联系人上传**

通过上传用户的联系人数据，可以使得人名的识别更加准确，联系人上传代码示例如下：

```
BDVRDataUploader *contactsUploader = [[BDVRDataUploader alloc] initDataUploader:sel
f];
self.contactsUploader = contactsUploader;
[contactsUploader release];
[self.contactsUploader setApiKey:@"apiKey" withSecretKey:@"secretKey"];
NSString *jsonString = @"[{\"name\": \"test\",\"frequency\": 1},{\"name\": \"releas
e\",\"frequency\": 2}]";
[self.contactsUploader uploadContactsData:[jsonString dataUsingEncoding:NSUTF8Stri
ngEncoding]];
```

接口详细信息请参考头文件。上传数据格式必须跟示例中一样。

**请求联系人识别结果**

用户上传联系人数据之后，可以使得人名的识别更加准确，前提是需要告知服务器请求之前上传过的联系人，以免在不需要的情况下对识别结果造成不期望的影响，请求联系人识别结果需要通过 BDVoiceRecognitionClient.setConfig 接口进行控制，代码如下：

```
[[BDVoiceRecognitionClient sharedInstance] setConfig:@"enable_contacts" withFlag:YE
S];
```

如果使用了 SDK 提供的 UI 对话框，将传入的 BDRecognizerViewParamsObject 对象的 enableContacts 参数置为 YES 即可：

```
BDRecognizerViewParamsObject *paramsObject = [[BDRecognizerViewParamsObject alloc]
init];
paramObject.enableContacts = YES;
// 其他参数配置
// ...
// 调起对话框，开始识别
```

# 5 注意事项

1. BDVRClient 使用前必须在百度开放云平台中注册使用权限。

2. BDVRClient 采用单例模式，因此，开发者在使用语音识别功能时，应避免创建多个对象，否则会造成识别失败。开发者要严格使用接口文件中获取方法，不要使用 init。

3. 如果开启播放提示音功能时使用自己的语音文件，格式必须是 caf，并且直接覆盖识别库中的资源文件，不要修改文件名和文件路径，否则 BDVRClient 无法进行播放。

4. 如果在一次语音识别过程中不使用监听语音音量功能（之前曾开启过），注意取消监听，以免造成不必要的 CPU 消耗。

5. 开始识别时间：开发者在使用 API 方式进行识别时，也需要注意语音识别真正开始的时间（即应用程序收到 EVoiceRecognitionClientWorkStatusStartWorkIng 通知的时刻），在收到此通知前不应在界面上暗示用户可以说话。

# FAQ

1. 如何联系我们，和反馈意见？

   如果在程序开发中遇到了 Bug，或者在使用过程中有好的建议，请发送邮件至 voice_feedback@baidu.com

   欢迎加入百度语音开发者交流 Hi 群：1412729 或 QQ 群：369510575 获取更多技术支持